IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	David C. Collins	Examiner: Jeffrey S. Smith
Serial No.:	10/821,130	Group Art Unit: 2624
Filed:	April 8, 2004	Docket No.: 200400519-1
Title:	GENERATING AND DISPLAYING SPATIALLY OFFSET SUB-FRAMES	

DECLARATION OF PRIOR INVENTION UNDER 37 C.F.R. § 1.131

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir/Madam:

This Declaration is submitted to establish invention of the subject matter of the present patent application prior to the publication date of February 12, 2004 of U.S. Patent Application No. US 2004/0027363 (hereinafter referred to as "Allen").

1. The person making this Declaration is inventor David C. Collins.
2. Accompanying this Declaration are Exhibit A and Exhibit B to establish reduction to practice of the subject matter of the present patent application in the United States prior to February 12, 2004.
3. Exhibit A (10 pages) includes a Hewlett-Packard Company (HP) Invention Disclosure and accompanying attachment (hereinafter referred to collectively as the "Disclosure").
4. Exhibit B is a page from a Laboratory Notebook of inventor David C. Collins (hereinafter referred to as the "Laboratory Notebook Page").
5. The Disclosure and the Laboratory Notebook Page were prepared by the inventor in the United States prior to February 12, 2004.
6. The Disclosure and the Laboratory Notebook Page were witnessed in the United States prior to February 12, 2004.
7. The Disclosure was submitted by the inventor to the HP Legal Department in the United States prior to February 12, 2004.
8. The Disclosure was assigned HP Patent Disclosure No. 200400519.

9. The Disclosure and the Laboratory Notebook Page describe the subject matter of the present patent application – i.e., the subject matter of independent claims 1, 11, 20, and 25 – as follows.

Claims	Disclosure & Laboratory Notebook Page
Claim 1: "A method of displaying an image with a display device, the method comprising:" Claim 11: "A system for displaying an image, the system comprising:" Claim 20: "A system for generating sub-frames for display at spatially offset positions to generate the appearance of an image, the system comprising:" Claim 25: "A computer-readable medium storing computer-executable instructions, which, when executed by a computer processing system, cause the system to perform a method of generating a sub-frame image which comprises a plurality of sub-frames for display at spatially offset positions to generate the appearance of an image, comprising:"	The Disclosure (pp. 1-10) describes the generation of sub-frames for display at spatially offset positions to form a displayed image. Wobulation, as referenced on pp. 1, 2, and 4 of the Disclosure, refers to the display of sub-frames at spatially offset positions to display an image using a display device. The Laboratory Notebook Page shows pictures labeled "Center Adaptive w/ History" and "Adaptive w/ History" that each represent a displayed image.
Claim 1: "receiving image data for the image;" Claim 11:	Image data is shown on p. 6 of the Disclosure (a region of interest "ROI" for a given sub-frame pixel value is shown relative to the

"a buffer adapted to receive image data for the image;" Claim 20: "means for receiving image data corresponding to the image;" Claim 25: "receiving image data corresponding to the image;"	image data). Image data is operated on by the set of computer-executable instructions listed on pp. 8-10 of the Disclosure. The Laboratory Notebook Page shows a picture labeled "Original" that represents received image data.
Claim 1: "generating first and second sub-frames to include first and second sub-frame pixel values, respectively, and wherein the first sub-frame pixel value is calculated using the image data and the second sub-frame pixel value;" Claim 11: "an image processing unit configured to generate first and second sub-frames comprising a plurality of rows of sub-frame pixel values, wherein each of the sub-frame pixel values in each of the plurality of rows is calculated using the image data and at least one sub-frame pixel value from a previous one of the plurality of rows;" Claim 20: "means for generating a plurality of rows of initial values using the image data;	On p. 4 of the Disclosure, "[a]ll of the sub-frames are processed together at the same time, and the sub-frames are intertwined." The diagram on p. 4 of the Disclosure illustrates how the sub-frames are intertwined in one embodiment. Pp. 6-7 of the Disclosure describe the generation of final sub-frame pixel values for the sub-frames. As shown in the diagrams and described on pp. 6-7 of the Disclosure, each final sub-frame pixel value is generated using values from a region of interest. As shown in the diagrams on p. 6 and p. 7 of the Disclosure, the region of interest for the second and subsequent rows of image data includes final sub-frame values, i.e., "Final values calculated". See also p. 6 ("[m]any of the final sub-frame values will already have their final values computed.")

means for accessing a row of history values generated using the image data; and means for generating a sub-frame pixel value using the row of history values and the plurality of rows of initial values." <u>Claim 25:</u> "generating a first plurality of initial values associated with a first pixel which corresponds to a first one of the plurality of sub-frames using the image data; generating a first sub-frame pixel value using the image data and the first plurality of initial values, wherein the first sub-frame pixel value comprises a first history value; generating a second plurality of initial values associated with a second pixel which corresponds to a second one of the plurality of sub-frames using the image data; and generating a second sub-frame pixel value using the image data, the second plurality of initial values, and the first history value."	As shown in the diagrams on p. 6 and p. 7 of the Disclosure, final sub-frame pixel values for the second and subsequent rows in the diagram on p. 6 are generated using final sub-frame pixel values from the previous row in the region of interest as shown in the diagram on p. 7. The final sub-frame pixel values (e.g., the previous row) used to generate a given final sub-frame pixel value are also referred to as history values. See Disclosure diagram on p. 8 ("Historic Values"). The "Algorithm Steps" on p. 8 of the Disclosure show the generation of rows of initial values. The set of computer-executable instructions listed on pp. 8-10 of the Disclosure performs a method of generating a final sub-frame pixel value (guess1_0M) using other final sub-frame pixel values (e.g., final0_5M, final0_4M, final0_3M, final0_2M, final0_1M, and final0_0M,) and initial values (e.g., image1_1M to image1_5M, image2_1M to image2_5M, image3_1M to image3_5M, image4_1M to image4_5M). The "Algorithm Steps" on the Laboratory Notebook Page show the generation of rows of initial values.
<u>Claim 1:</u> "alternating between displaying the first sub-	Wobulation, as referenced on pp. 1, 2, and 4 of the Disclosure, refers to the display of sub-

frame, including displaying the first sub-frame pixel value, in a first position and displaying the second sub-frame, including displaying the second sub-frame pixel value, in a second position spatially offset from the first position." <u>Claim 11:</u> "a display device adapted to alternately display the first sub-frame in a first position and the second sub-frame in a second position spatially offset from the first position."	frames at spatially offset positions using a display device.

10. The subject matter of the present patent application was tested as follows.

   a. The set of computer-executable instructions is listed on pp. 8-10 of the Disclosure embodies the method of generating a final sub-frame pixel value described in the Disclosure.

   b. The set of computer-executable instructions listed on pp. 8-10 of the Disclosure was written prior to February 12, 2004.

   c. The set of computer-executable instructions listed on pp. 8-10 of the Disclosure was executed with at least one test image on at least one computer system to generate final sub-frame pixel values for sub-frames for the test image prior to February 12, 2004.

   d. The sub-frames generated for the test image were displayed on at least one display device to confirm the successful generation of sub-frames and the successful reproduction of the test image prior to February 12, 2004. Laboratory Notebook Page (pictures labeled "Center Adaptive w/ History" and "Adaptive w/ History").

11. The execution of the set of computer-executable instructions listed on pp. 8-10 of the Disclosure on the computer system to generate the final sub-frame pixel values for the sub-frames combined with the display of the sub-frames on the display device demonstrates actual working conditions or a realistic simulation of working conditions for the subject matter of the present patent application.

12. The successful generation of sub-frames and successful reproduction of the test image demonstrates utility beyond a probability of failure for the subject matter of the present patent application.

13. The test results for the subject matter of the present patent application, i.e., the successful generation of sub-frames and successful reproduction of the test image, are reproducible by executing the set of computer-executable instructions listed on pp. 8-10 of the Disclosure on a suitable computer system with another suitable image to generate final sub-frame values for sub-frames for the image and displaying the sub-frames on a suitable display device to reproduce the image.

14. As demonstrated by this Declaration, Exhibit A, and Exhibit B, the subject matter of the present patent application was reduced to practice in the United States prior to the publication date of February 12, 2004 of Allen.

As a person signing below, I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Signed: David C Collins　　　Date: 2008-Oct-13
David C. Collins

hp invent

200400519: A Practical Implementati...

[redacted]

Innovation Number
200400519

[redacted]

Disclosure Summary

Title A Practical Implementation of Multipass Adaptive Wobulation

Abstract Optimal subframe generation for wobulation can be implemented with an interative algorithm. This disclosure describes a practical method for implementing multiple passes of the wobulation subframe generation algorithm.

Attachments **Adaptive_Kernel_With_History2.doc [188928 bytes] -** [redacted]

Inventors David C Collins

Invention Disclosures

hp invent

Disclosure No. 200400519

[redacted]

PD No.	Date/Time Submitted	Collection
200400519	[redacted]	[redacted]

The information contained in this document is **HP CONFIDENTIAL** and may not be disclosed to others without prior authorization. Submit this disclosure to the HP Legal Department as soon as possible. No patent protection is possible until a patent application is authorized, prepared, and submitted to the Government.

General Information

Title	A Practical Implementation of Multipass Adaptive Wobulation
Abstract	Optimal subframe generation for wobulation can be implemented with an interative algorithm. This disclosure describes a practical method for implementing multiple passes of the wobulation subframe generation algorithm.
Projects	[redacted]
Products	[redacted]

Attachments

Attachments	**Adaptive_Kernel_With_History2.doc [188928 bytes] -** [redacted]

Description of Invention

Problems Solved	Optimal subframe generation for wobulation is computationally complex. Niranjan at HP Labs has previously disclosed an iterative method for solving the optimal subframe generation problem. This

	has been called the adaptive wobulation method. Running multiple iterations of the adaptive algorithm is expensive in terms of memory bandwidth, and is not practical for a consumer grade product. This disclosure describes a method for computing multiple passes of the adaptive algorithm with limited memory bandwidth requirements.
Prior Solutions	Two straight forward methods exist for optimal subframe generation using the adaptive method. The first method is to simply run the algorithm for multiple iterations. This results in the following steps: 1) Create the initial subframe 2) Generate a simulated image (i.e. merge the subframes together) 3) Calculate the error by subracting the simulated image from the original image 4) Feedback the error to update the subframes goto step 2 This method requires a great deal of memory bandwidth as the simulated image must be stored in memory, and the subframes must be stored in memory for each iteration. The second method is to apply the iterative algorithm on a portion of the image, and then calculate the final subframe value in a single step. Once a value have been calculated, the region of interest is shifted and a new value is calculated. This method requires a small amount of memory bandwidth, but many calculations are redundant, and the internal memory requirements of the calculation hardware are still quite large. The region of interest grows as the number of interations grows. Thus, to date, only one pass of the adaptive algorithm has been seriously proposed.
Description	The description is given in the attached document. In summary, the invention reduces the number of rows of data that are required for the multipass adaptive wobulation algorithm. This is accomplished by making a small compromise for some of the initial values, and by keeping track of the previous row of final subframe values.
Advantages	This invention enables multipass wobulation to be implemented in an ASIC without increasing the memory bandwith requirements. Also, the amount of on chip memory is kept relatively small. The techniques in this invention can be applied in a straight forward fashion for the simplified center adaptive algorithm that was recently disclosed.

Invention History

Published	[redacted]
Announced	
Disclosed	
Next Three Months	
Described	
Built	
Government Contract	
Related Disclosure	
Innovation Workshop	

Inventor Information

Inventors	**David C Collins** Hewlett-Packard Company Corvallis [redacted]

Witnesses

Witnesses	**Richard Aufranc** Hewlett-Packard Company Corvallis [redacted]

Classification

Recommended Classification	[redacted]
Keywords	
Recommended Merlin Entity	
Recommended Merlin Loc	
Recommended Merlin Responsible_attorney	

Administrative Record

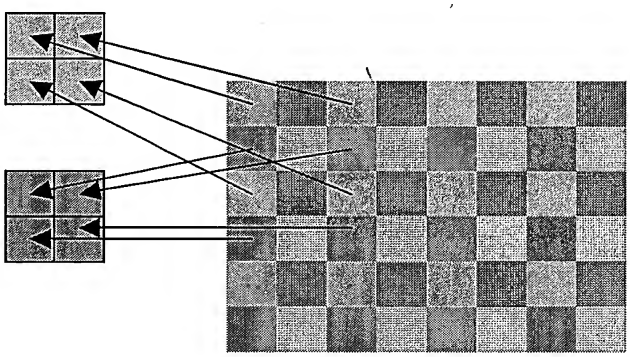
Date/Time Submitted	[redacted]
PD Number	200400519
Date PD Number Assigned	[redacted]

Filing Decisions

[redacted]

Adaptive Kernel With History

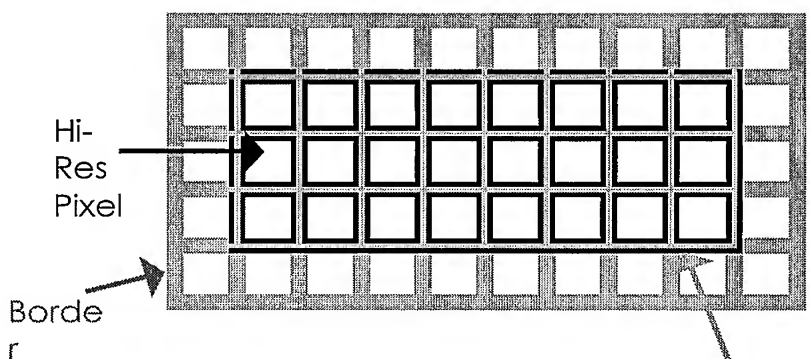
The following explanation is assuming 4 position wobulation. Thus, four low resolution subframes must be generated subframes – one for each image position. All of the subframes are processed together at the same time, and the subframes are intertwined. Thus, every four pixel will correspond to a different subframe. The following diagram illustrates the idea.



To compute the adaptive algorithm, subframes are computed, a simulated image (the four subframes merged together) is computed, an error image is computed. The error is averaged, and then feedback into the subframes. This process is repeated iteratively, and within a few iterations, this method converges to the solution. The standard adaptive algorithm is briefly defined below:

Simulation Kernel

$$(K_s) \quad \begin{matrix} \frac{1}{4} & \frac{1}{4} & 0 \\ \frac{1}{4} & \frac{1}{4} & 0 \\ 0 & 0 & 0 \end{matrix}$$



Error Kernel

$$(K_e) \quad \begin{matrix} 0 & 0 & 0 \\ 0 & \frac{1}{4} & \frac{1}{4} \\ 0 & \frac{1}{4} & \frac{1}{4} \end{matrix}$$

Iterative Algorithm

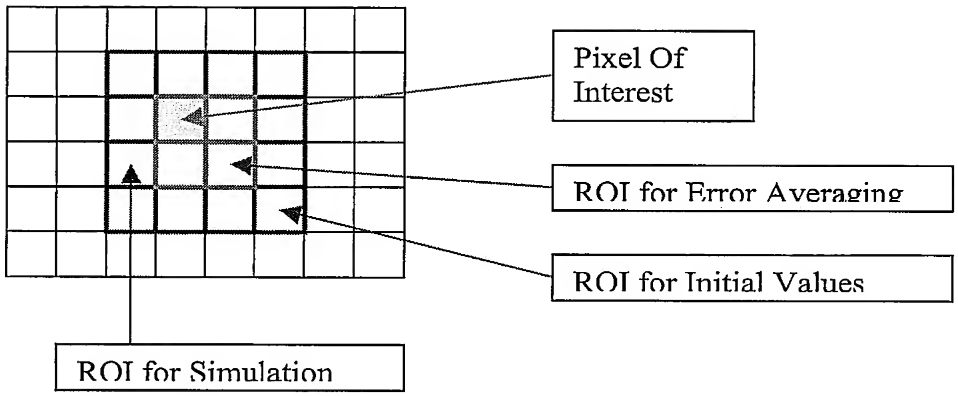
Simulation = K_s * Subframes

Error = Image – Simulation

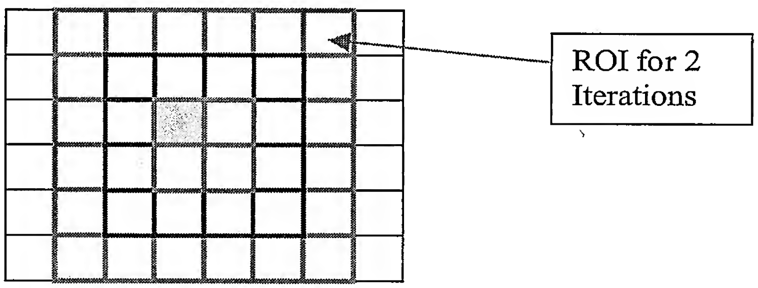
$Error_{avg}$ = K_e * Error

Subframes = Subframes + α x $Error_{avg}$

In order to calculate the subframe value for the a single pixel a 4x4 ROI is required for the one iteration. The diagram below illustrates the required pixel values.

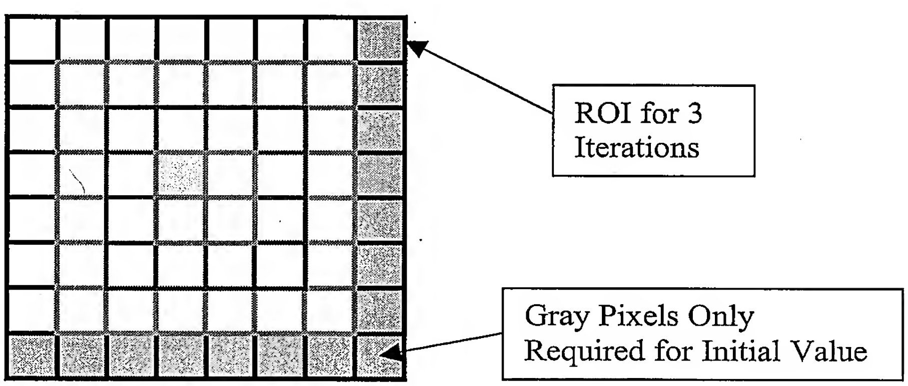


For two iterations, a 6x6 ROI is required.



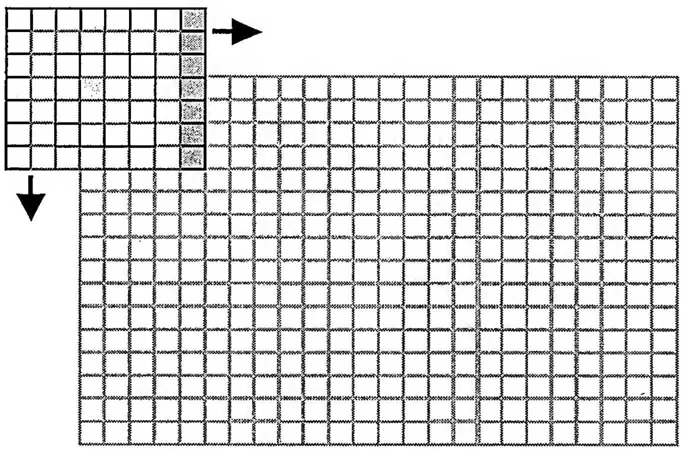
For 3 iterations an ROI of 8x8 is needed, and in general for n iterations an ROI of (2n+2)x(2n+2) is required.

Consider 3 iterations of the adaptive algorithm. To compute the optimal adaptive algorithm, an 8x8 ROI is required.

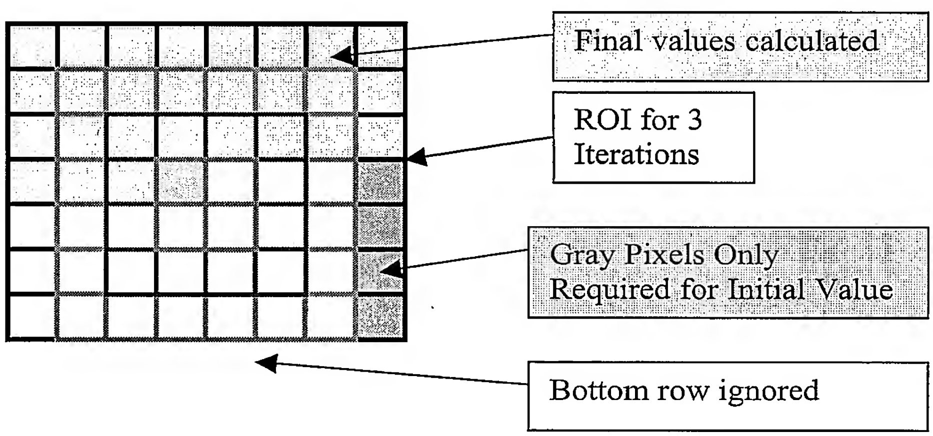


The bottom row of the ROI is only required to generate the initial guess for the pixel values that are at least 2 pixels away from the primary pixel of interest. There is very little impact in eliminating the last row from the ROI. This just implies that for some of the pixels a simplified calculation will have to be used for the initial subframe value.

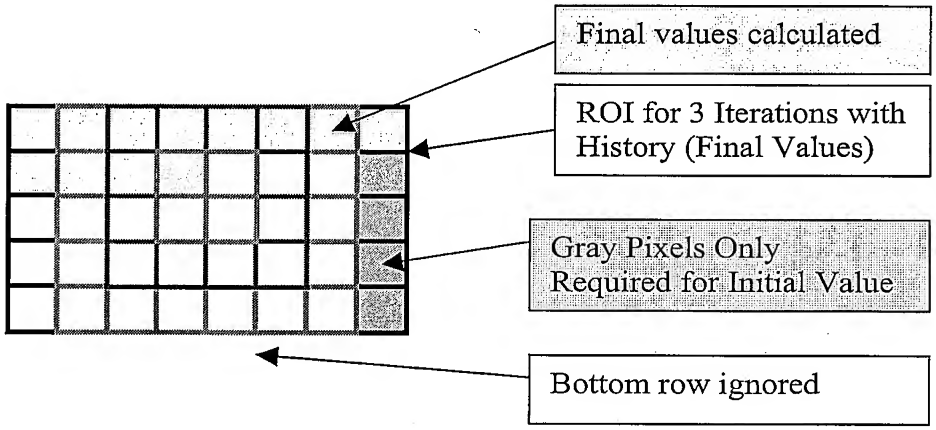
To generate the subframe values, the region of interest is positioned such that the pixel of interest is located over each pixel location. Typically, the ROI starts in the upper left corner, and the image is processed in a raster format. Thus, the first row of subframe values are calculated, followed by the second, and then the third row of data. The following image illustrates how the ROI is moved across the image and the subframe values are generated.



Since the image processing is typically done in a raster pattern. Many of the final subframe values will already have their final values computed. The diagram below illustrates the pixel locations with final values.



Now if we look at the calculations we can see that instead of using three pixel rows of image data above the pixel of interest, we can substitute the final calculated values instead. Thus, we can compute the 3-pass adaptive algorithm by using 1 row of the previously calculated subframe values along with 4 rows of the original image. This results in the following ROI:



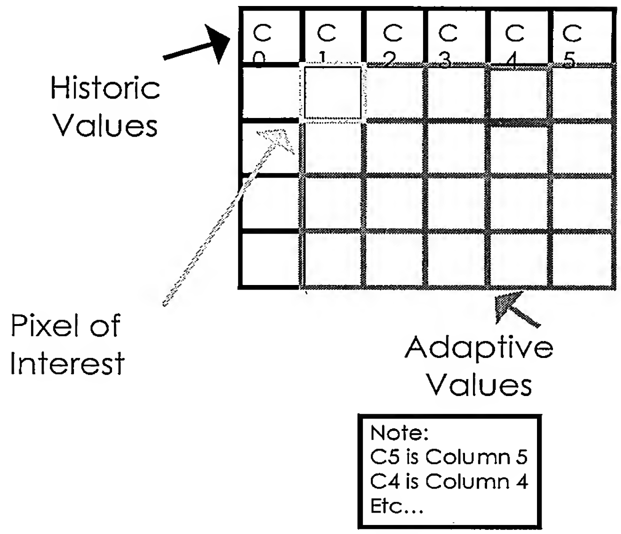
This same approach can be used for a variety of iterative or adaptive algorithms. Essentially two main simplifications have been done. First, instead of using a number of image rows above the pixel of interest, only one row is used, and this row contains the calculated subframe values and not the original image values. The second simplification comes from truncating the kernel in heighth, but not in width. It should also be obvious that these techniques can be applied to achieve more than 3 iterations. Three iterations was used for the example because it highlights the novelties of the invention, and three is about the right number of iterations to achieve almost all of the benefits of the iterative algorithm.

The benefits of this algorithm are best appreciated by an ASIC implementation. With this method the ASIC could be designed to include enough onboard memory to store the original image rows and one row of history (calculated subframe values). The image would be processed as it enters the ASIC. Only the final subframe values would need to be stored in external memory, and then read out at the proper time. Thus the memory bandwidth only has to support the writing and reading of a full frame of data. Also, as the kernel grows in height, more memory would be required on the ASIC. This explains why the goal was to reduce the number of rows needed by the ROI.

In summary, the 3-pass adaptive kernel with history can be implemented by a relatively simply algorithm. The following steps are performed once per pixel location.

Algorithm Steps

1 Calculate Adaptive C_4 (Initial Guess)
2 Calculate Sim C_4 (Convolve with Ks)
3 Calculate Error C_4 (Convolve with Ke)
4 Calculate Sim C_3
5 Calculate Error C_3
6 Calculate Adaptive C_3 ($x = x + \alpha$ Error)
7 Calculate Sim C_2
8 Calculate Error C_2
9 Calculate Adaptive C_2
10 Calculate Sim C_1
11 Calculate Error C_1



And for the truly inspired, the 3 pass adaptive algorithm can be implement with the following C++ code. The same approach has also been applied to the simplified center adaptive algorithm:

```
unsigned char AdaptiveStandardKernel::Calculate
(       unsigned char final0,
        unsigned char image1,
        unsigned char image2,
        unsigned char image3,
        unsigned char image4
)
{
        int temp;
        int x,y;        // temporary values to explicitly reduce the number of calculations

        shiftValues();

        final0_5M = final0;
        image1_5M = image1;
        image2_5M = image2;
        image3_5M = image3;
        image4_5M = image4;

        // calculate guess for column 4 -> average of 4 image pixels
        x         = image1_4M+image1_5M;
        y         = image2_4M+image2_5M;
        guess1_4M = (x+y)>>2;
        x         = image3_4M+image3_5M;
        guess2_4M = (x+y)>>2;
        y         = image4_4M+image4_5M;
        guess3_4M = (x+y)>>2;
        guess4_4M = y>>1;

        // compute sim column 4
        x          = final0_3M+final0_4M;
        y          = guess1_3M+guess1_4M;
        int sim1_4 = x+y;
        x          = guess2_3M+guess2_4M;
        int sim2_4 = x+y;
```

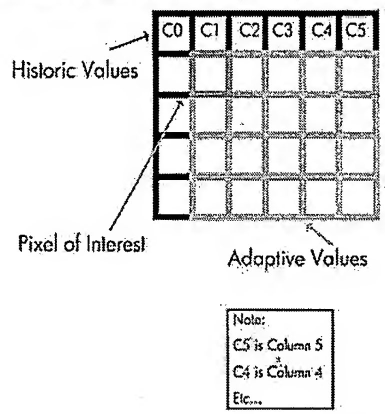
```
y           = guess3_3M+guess3_4M;
int sim3_4 = x+y;
x           = guess4_3M+guess4_4M;
int sim4_4 = x+y;

int err1_4  = (image1_4M<<2) - sim1_4;       // column 4
int err2_4  = (image2_4M<<2) - sim2_4;
int err3_4  = (image3_4M<<2) - sim3_4;
int err4_4  = (image4_4M<<2) - sim4_4;

// compute sim column 3
x           = final0_2M+final0_3M;
y           = guess1_2M+guess1_3M;
int sim1_3 = x+y; // column 3
x           = guess2_2M+guess2_3M;
int sim2_3 = x+y;
y           = guess3_2M+guess3_3M;
int sim3_3 = x+y;
x           = guess4_2M+guess4_3M;
int sim4_3 = x+y;

int err1_3  = (image1_3M<<2) - sim1_3;       // column 3
int err2_3  = (image2_3M<<2) - sim2_3;
int err3_3  = (image3_3M<<2) - sim3_3;
int err4_3  = (image4_3M<<2) - sim4_3;

// compute guess column 3
// divide by 4 for average error, 4 for simulation, & 4 for numerator
x      = err1_3+err1_4;
y      = err2_3+err2_4;
temp   = x+y;                          // guess 1_3
temp   = (temp * alpha1M) >> 6;
temp   = temp + guess1_3M;
guess1_3M = max(0,min(temp,255));      // clip value

x      = err3_3+err3_4;
temp   = x+y;                          // guess 2_3
temp   = (temp * alpha1M) >> 6;
temp   = temp + guess2_3M;
guess2_3M = max(0,min(temp,255));      // clip value

y      = err4_3+err4_4;
temp   = x+y;                          // guess 3_3
temp   = (temp * alpha1M) >> 6;
temp   = temp + guess3_3M;
guess3_3M = max(0,min(temp,255));      // clip value

temp   = y;                            // guess 4_3
temp   = (temp * alpha1M) >> 5;        // by 5 since x is not available
temp   = temp + guess4_3M;
guess4_3M = max(0,min(temp,255));      // clip value

x           = final0_1M+final0_2M;
y           = guess1_1M+guess1_2M;
int sim1_2 = x+y; // column 2
x           = guess2_1M+guess2_2M;
int sim2_2 = x+y;
y           = guess3_1M+guess3_2M;
int sim3_2 = x+y;
x           = guess4_1M+guess4_2M;
int sim4_2 = x+y;

int err1_2  = (image1_2M<<2) - sim1_2;       // column 2
int err2_2  = (image2_2M<<2) - sim2_2;
int err3_2  = (image3_2M<<2) - sim3_2;
int err4_2  = (image4_2M<<2) - sim4_2;

x      = err1_2+err1_3;
y      = err2_2+err2_3;
temp   = x+y;                          // guess 1_2
temp   = (temp * alpha2M) >> 6;
```

```
    temp   = temp + guess1_2M;
    guess1_2M = max(0,min(temp,255));       // clip value

    x      = err3_2+err3_3;
    temp   = x+y;                           // guess 2_2
    temp   = (temp * alpha2M) >> 6;
    temp   = temp + guess2_2M;
    guess2_2M = max(0,min(temp,255));       // clip value

    y      = err4_2+err4_3;
    temp   = x+y;                           // guess 3_2
    temp   = (temp * alpha2M)>>6;
    temp   = temp + guess3_2M;
    guess3_2M = max(0,min(temp,255));       // clip value

    temp   = y;                             // guess 4_2
    temp   = (temp * alpha2M)>>5;
    temp   = temp + guess4_2M;
    guess4_2M = max(0,min(temp,255));       // clip value

    x           = final0_0M+final0_1M;
    y           = guess1_0M+guess1_1M;
    int sim1_1 = x+y; // column 1
    x           = guess2_0M+guess2_1M;
    int sim2_1 = x+y;
    y           = guess3_0M+guess3_1M;
    int sim3_1 = x+y;
    x           = guess4_0M+guess4_1M;
    int sim4_1 = x+y;

    int err1_1  = (image1_1M<<2) - sim1_1;        // column 1
    int err2_1  = (image2_1M<<2) - sim2_1;
    int err3_1  = (image3_1M<<2) - sim3_1;
    int err4_1  = (image4_1M<<2) - sim4_1;

    x      = err1_1+err1_2;
    y      = err2_1+err2_2;
    temp   = x+y;                           // guess 1_1
    temp   = (temp * alpha3M)>>6;
    temp   = temp + guess1_1M;
    guess1_1M = max(0,min(temp,255));       // clip value

    x      = err3_1+err3_2;
    temp   = x+y;                           // guess 2_1
    temp   = (temp * alpha3M)>>6;
    temp   = temp + guess2_1M;
    guess2_1M = max(0,min(temp,255));       // clip value

    y      = err4_1+err4_2;
    temp   = x+y;                           // guess 3_1
    temp   = (temp * alpha3M)>>6;
    temp   = temp + guess3_1M;
    guess3_1M = max(0,min(temp,255));       // clip value

    temp   = y;                             // guess 4_1
    temp   = (temp * alpha3M)>>5;
    temp   = temp + guess4_1M;
    guess4_1M = max(0,min(temp,255));       // clip value

    return guess1_0M;
}
```

Adaptive Kernel w/ History

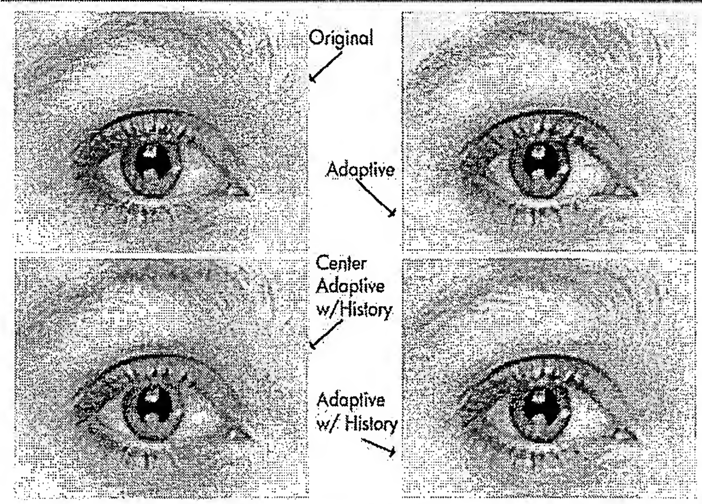
Algorithm Steps

1 Calculate Adaptive C_4 (Initial Guess)
2 Calculate Sim C_4 (Convolve with Ks)
3 Calculate Error C_4 (Convolve with Ke)
4 Calculate Sim C_3
5 Calculate Error C_3
6 Calculate Adaptive C_3 ($x = x + \alpha$ Error)
7 Calculate Sim C_2
8 Calculate Error C_2
9 Calculate Adaptive C_2
10 Calculate Sim C_1
11 Calculate Error C_1
12 Calculate Adaptive C_1
13 Shift Values Left & Go to Step 1



11/6/2003 David Collins HP Confidential 9

Comparison of Algorithms



11/6/2003 David Collins HP Confidential 10

Adaptive Simulations

- All of the adaptive simulations were done with the following parameters:
  - Alpha = 2.0
  - Iterations = 3
  - Initial guess for the standard adaptive algorithm was an average of 4 pixels
  - Initial guess for the center adaptive algorithm was pixel extraction

Adaptive Algorithm Observations

- The 3 Pass Adaptive algorithm can be calculated with 4 image lines & 1 history line.
- Pixel Extraction looks better than the Adaptive Algorithm on Single Pixel Features.
- The Adaptive Algorithm looks better than Pixel Extraction on Natural Images.
- The "Center" Adaptive Algorithm looks good on both Single Pixel Features and Natural Images.
- A Simplified "Center" Adaptive Algorithm greatly reduces the algorithm complexity and produces equivalent results.

Recorded by

Invented By David C Collins [redacted]

11/21/03